**Сравнение JUnit 5 vs TestNG**

Содержание

[1. Архитектура 1](#_Toc94730520)

[1.1 Настройка теста 2](#_Toc94730521)

[2 Аннотации 2](#_Toc94730522)

[3. Совместное выполнение тестов 4](#_Toc94730523)

[4. Тестирование исключений 5](#_Toc94730524)

[5. Параметризованные тесты 5](#_Toc94730525)

[6. Тайм-аут теста 7](#_Toc94730526)

[7. Пользовательские имена тестов 7](#_Toc94730527)

[8. Assertions 8](#_Toc94730528)

[9. Assumptions 8](#_Toc94730529)

[10. Параллельное выполнение 9](#_Toc94730530)

[11. HTML Reports 9](#_Toc94730531)

[12. Listeners 10](#_Toc94730532)

[Вывод 10](#_Toc94730533)

### Архитектура

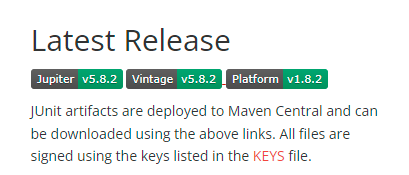
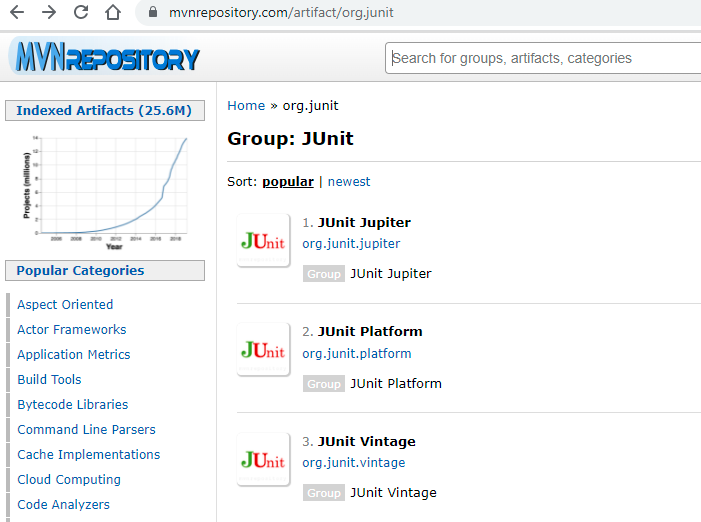
JUnit 5

JUnit 5 = Платформа JUnit + JUnit Jupiter + JUnit Vintage

В проекте Maven в файл pom.xml необходимо подключить 3 зависимости.

Адрес для скачивания:

<https://junit.org/junit5/> или https://mvnrepository.com/artifact/org.junit

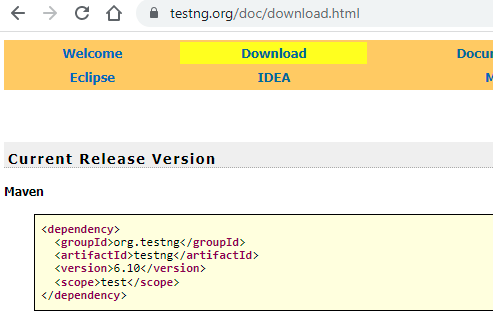
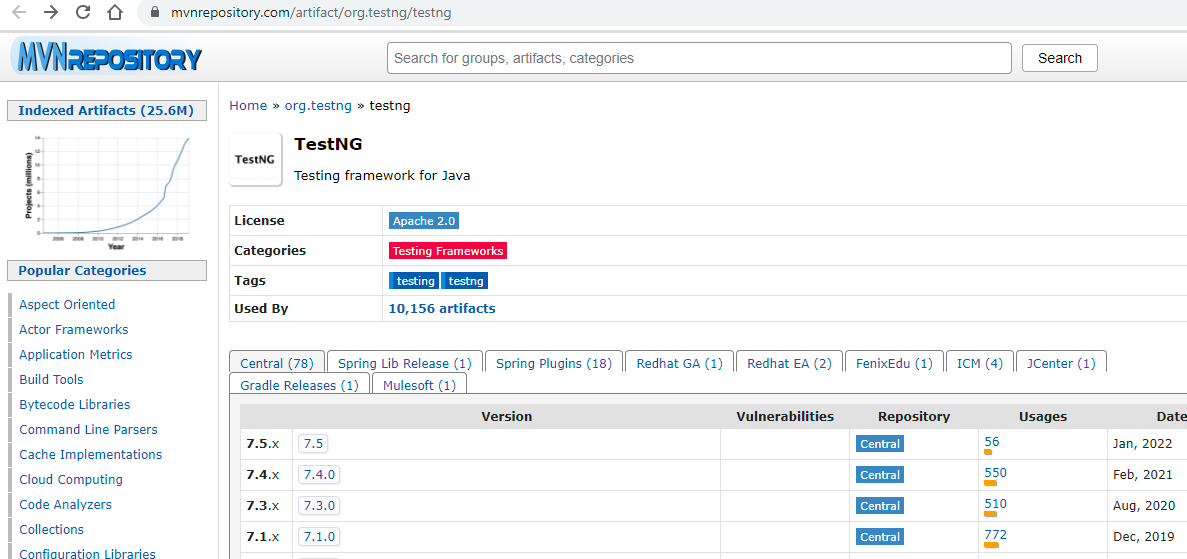
 

TestNG

В проекте Maven в файл pom.xml необходимо подключить одну зависимость.

Адрес для скачивания:

<https://testng.org/doc/download.html> или https://mvnrepository.com/artifact/org.testng/testng

**Преимущество на стороне TestNG, т.к. он более легковесный**.

### 1.1 Настройка теста

**TestNG**

Информация о тесте записывается в файл testng.xml. Набор тестов (Test Suite) представлен XML-файлом, это позволяет выполнять гибкую настройку тестов.   
Атрибуты Test Suite

|  |  |
| --- | --- |
| Name | Название комплекта, это обязательный атрибут |
| Verbose | Уровень детализации операции, уровень 0-10, из которых 10 наиболее подробный |
| Parallel | Запускает ли TestNG разные потоки для запуска этого пакета, по умолчанию - none, другие уровни - методы, тесты, классы, экземпляры. |
| thread-count | Количество используемых потоков, если включен параллельный режим (без учета других методов). аннотации используют тип аннотации в тесте. |

Нюансы TestNG конфигурации:

* Test Suite включает в себя 1 или более тестов
* Тест может быть набором групп, пакетов, классов, тестовых методов.
* Помимо полных названий можно использовать паттерны при указании имен групп, классов и т.д.
* В одном файле конфигурации может быть только 1 suite.
* Файлов testng.xml может быть много.
* Можно задавать параметры на уровне конфига и затем использовать их в тестовых методах через @Parameters.

JUnit 5

XML-файл с набором тестов отсутствует.

### 2 Аннотации

TestNG И JUnit 5 тестирование основано на аннотациях, и они похожи по своей природе и поведению.

|  |  |  |
| --- | --- | --- |
| Аннотация | **Junit 5** | **TestNG** |
| Аннотация теста | @Test | @Test |
| Запуск перед suite | — | @BeforeSuite |
| Запуск после suite | — | @AfterSuite |
| Запуск перед тестированием | — | @BeforeTest |
| Запуск после тестирования | — | @AfterTest |
| Запуск перед тестом из группы | — | @BeforeGroups |
| Запуск после теста из группы | — | @AfterGroups |
| Запуск перед классом | @BeforeAll | @BeforeClass |
| Запуск после класса | @AfterAll | @AfterClass |
| Запуск перед каждым тестовым методом | @BeforeEach | @BeforeMethod |
| Запуск после каждого тестового метода | @AfterEach | @AfterMethod |
| Игнорировать тест | @Disabled | @Test(enbale=false) |
| Test factory для динамических тестов | @TestFactory | @Factory |
| Тегирование | @Tag | — |
| Таймаут | @Test(timeout = 1000) | @Test(timeout = 1000) |

Описание аннотаций JUnit 5

|  |  |
| --- | --- |
| **Аннотации** | **Описание** |
| @BeforeEach | Аннотированный метод будет запускаться перед каждым тестовым методом в тестовом классе. |
| @AfterEach | Аннотированный метод будет запускаться после каждого тестового метода в тестовом классе. |
| @BeforeAll | Аннотированный метод будет запущен перед всеми тестовыми методами в тестовом классе. Этот метод должен быть статическим. |
| @AfterAll | Аннотированный метод будет запущен после всех тестовых методов в тестовом классе. Этот метод должен быть статическим. |
| @Test | Он используется, чтобы пометить метод как тест junit. |
| @DisplayName | Используется для предоставления любого настраиваемого отображаемого имени для тестового класса или тестового метода |
| @Disable | Он используется для отключения или игнорирования тестового класса или тестового метода из набора тестов. |
| @Nested | Используется для создания вложенных тестовых классов |
| @Tag | Пометьте методы тестирования или классы тестов тегами для обнаружения и фильтрации тестов. |
| @TestFactory | Отметить метод — это тестовая фабрика для динамических тестов. |

Описание аннотаций TestNG

|  |  |
| --- | --- |
| **Аннотации** | **Описание** |
| **@BeforeSuite** | Метод с этой аннотацией будет выполнен до любых тестов, описанных в тестовом наборе. |
| **@AfterSuite** | Метод  с такой аннотацией будет запущен после всех тестов из тестового набора TestNG. |
| **@BeforeTest** | Такие методы будут выполняться до каждой секции Test в наборе тестов. |
| **@AfterTest** | Методы будут запускаться после каждой секции Test в тестовом наборе. |
| **@BeforeGroups** | Метод с аннотацией @BeforeGroup будет запускаться 1 раз до выполнения любого тестового метода указанной группы. |
| **@AfterGroups** | Такой метод будет запускаться 1 раз после запуска всех тестовых методов из указанной группы. |
| **@BeforeClass** | @BeforeClass помеченный метод выполняется однажды до запуска любого тестового метода в определенном тестовом классе. |
| **@AfterClass** | Такой метод запускается после выполнения всех тестовых методов тестового класса. |
| **@BeforeMethod** | Эти методы выполняются до запуска каждого тестового метода. |
| **@AfterMethod** | Такие методы запускаются после выполнения каждого тестового метода. |
| **@DataProvider** | Помечает метод как предоставляющий данные для тестового метода. Этот метод возвращает двумерный массив Object [ ] [ ] в качестве данных. |
| **@Factory** | Указывает что метод является фабрикой, возвращающей массив объектов Object[ ]. Эти объекты будут в последствии использованы как тестовые классы фреймворком TestNG. Такой механизм нужен для запуска набора тест-кейсов с различными значениями. |
| **@Listeners** | Применимо к тестовым классам. Определяет массив тестовых классов прослушивателей, реализующих org.testng.ITestNGListener. Предназначена для отслеживания статуса выполнения и логирования. |
| **@Parameters** | Аннотация используется для передачи параметров тестовому методу. Эти значения параметров подставляются из файла конфигурации testing.xml во время выполнения. |
| **@Test** | Отмечает класс или метод как тестовый. Если используется на уровне класса, все public методы этого класса помечаются как тестовые методы. |

TestNG предлагает более широкий набор аннотаций, чем JUnit 5

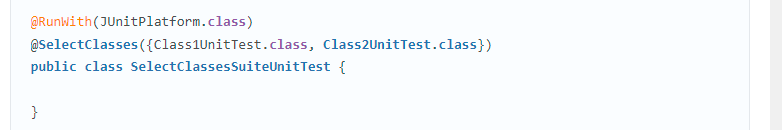
### 3. Совместное выполнение тестов

Совместное выполнение тестов в виде коллекции возможно как в JUnit 5, так и в TestNG, но они делают это по-разному.

JUnit 5

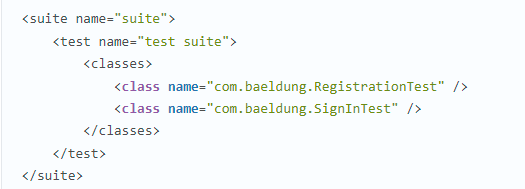
Используются аннотации @RunWith,@SelectPackages **и**@SelectClasses**для группировки тест-кейсов и запуска их как Test Suite. Если необходимо сгруппировать тест-кейсы различных пакетов для запуска вместе в Suite нужно использовать аннотацию @SelectPackages**



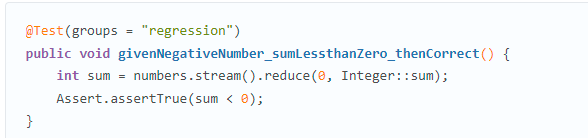


TestNG

Тесты группируются с помощью XML-файла (п.1.1)



Также может группировать методы с помощью аннотации @Test (groups = "groupName"):



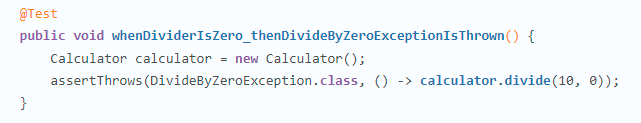
В целом тестирование для Test Suite лучше реализовано в TestNG.

### 4. Тестирование исключений

**Функция тестирования исключений с использованием аннотаций доступна как в** JUnit 5**, так и в** TestNG**.**

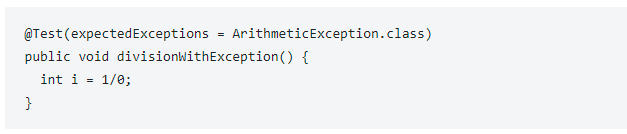
JUnit 5

Для тестирования исключений используется API assertThrows



TestNG

В аннотацию @Test добавляется параметр expectedExceptions



### 5. Параметризованные тесты

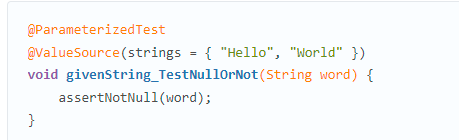
Повторное использование методов модульного тестирования с различным набором входных данных

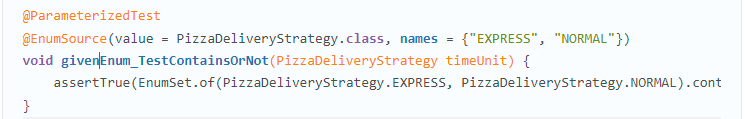
JUnit 5

**Методы тестирования используют аргументы данных непосредственно из настроенного источника**

По умолчанию JUnit 5 предоставляет несколько **source**-аннотаций, например:

* @ValueSource: use с массивом значений типа Short, Byte, Int, Long, Float, Double, Char и String:

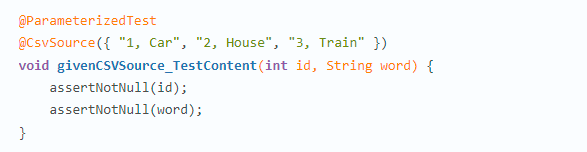


• @EnumSource — передает Enum-константы в качестве параметров методу тестирования:

• @MethodSource — передает (passes) внешние методы, генерирующие потоки:



* @CsvSource*—*использует значения CSV в качестве источника для параметров:



TestNG

Позволяет передавать параметры напрямую в методы тестирования двумя различными способами:

* с testng.xml
* с поставщиками данных

При использовании XML-файла метод тестирования аннотируется с помощью @Parameter:



Передача параметров с поставщиками данных

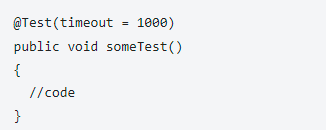
Когда нужно передать сложные параметры или параметры, которые необходимо создать из Java (сложные объекты, объекты, считанные из файла свойств или базы данных и т. Д.), Параметры можно передать с помощью поставщиков данных.

Поставщик данных — это метод, аннотированный @DataProvider . Эта аннотация имеет только один строковый атрибут: его имя. Если имя не указано, имя поставщика данных по умолчанию автоматически соответствует имени метода. Поставщик данных возвращает массив объектов.

### 6. Тайм-аут теста

Тайм-аут тестов означает, что тестовый пример должен завершиться неудачно, если выполнение не будет завершено в течение определенного указанного периода.

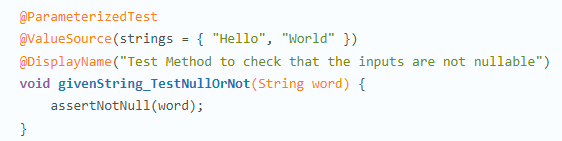
Для JUnit 5 **и** TestNG **синтаксис одинаковый.**



### 7. Пользовательские имена тестов

JUnit 5

Есть аннотация @DisplayName позволяет легко читать и понимать результаты тестирования



Каждый раз, при запуске теста, на выводе будет отображаться добавленное в аннотацию имя вместо имени метода.

TestNG

В настоящее время нет возможности указать пользовательское имя.

### 8. Assertions

Разный порядок реализации методов

JUnit 5

assertEquals(Object expected, Object actual, String message)

TestNG

assertEquals(Object actual, Object expected, String message)

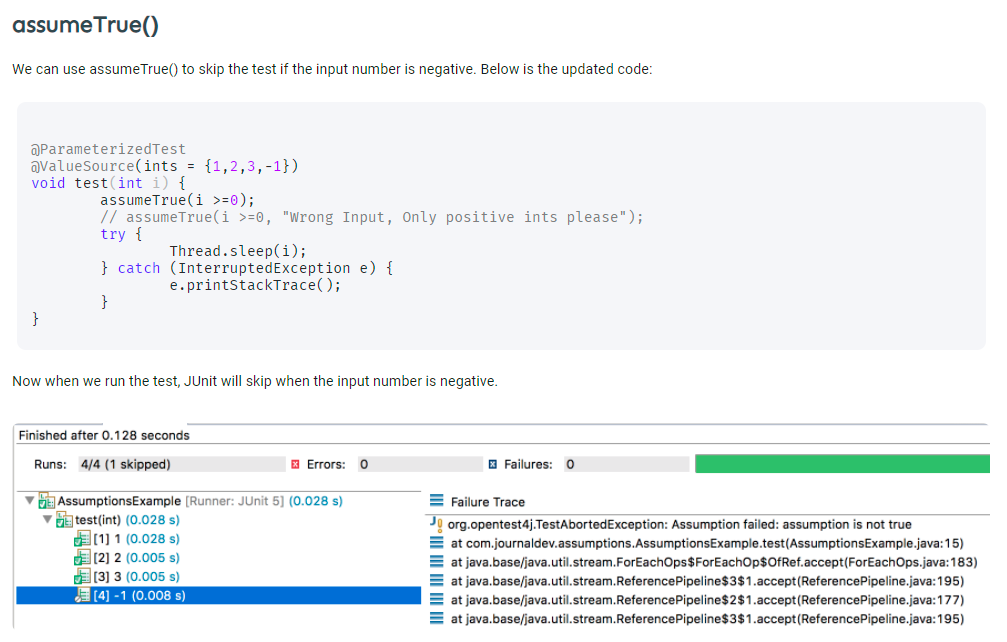
### 9. ****Assumptions****

JUnit 5

Класс Assumptions (предположения) предоставляет static методы для поддержки выполнения условного теста на основе предположений. Неуспешное предположение приводит к прерыванию теста.

Предположения обычно используются всякий раз, когда нет смысла продолжать выполнение данного метода тестирования. В отчете эти тесты будут отмечены как пройденные.





TestNG

В настоящее TestNG не поддерживает Assumptions.

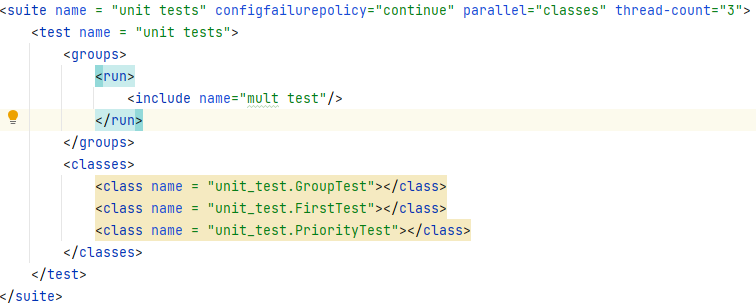
### 10. ****Параллельное выполнение****

JUnit 5

В настоящее время не поддерживает.

TestNG

Поддерживает. Реализовано через XML-файл



### 11. ****HTML Reports****

JUnit 5

Необходим дополнительный external plugin maven-surefire-report-plugin

TestNG

**Создается автоматически**

### 12. ****Listeners****

Основной задачей «Слушателей» (**Listeners**) является прослушивание определенного события и реагирование в соответствии с этим.

Основная цель, для которой слушатели используются программистами, заключается в создании журналов и создании пользовательских отчетов в соответствии с определенным сценарием.

JUnit 5

Нет механизма добавления **Listeners** через аннотации, работает только через Launcher API.

TestNG

Доступно много слушателей, и у каждого слушателя есть определенные методы, которые переопределяются.

**ITestListener** - один из наиболее часто используемых слушателей в Selenium Webdriver. Программист просто должен **реализовать интерфейс ITestListener** и переопределить все методы этого интерфейса, чтобы использовать его. Он делает вызов до и после каждого теста, присутствующего в комплекте.

* onStart: это первый и главный метод, который вызывается после создания экземпляра класса теста. Его также можно использовать для получения каталога, из которого выполняется тест.
* onFinish: это последний метод, вызываемый после выполнения всех переопределенных методов.
* onTestStart (результат ITestResult): вызывается каждый раз перед любым новым методом тестирования. Это указывает на то, что требуемый метод тестирования запущен.
* onTestFailure (результат ITestResult): вызывается, когда какой-либо метод теста не выполняется, поскольку он указывает на ошибки теста. Мы можем выполнять определенные задачи при неудачном тестировании, например, делать скриншот при сбое конкретного теста, чтобы получить более глубокое представление о сбое.
* onTestSkipped (результат ITestResult): вызывается, когда любой тестовый метод пропускается для выполнения.
* onTestSuccess (результат ITestResult): вызывается при успешном выполнении определенного метода тестирования.

### Вывод

В целом TestNG предоставляет больше возможностей для выполнения автоматизированного тестирования.

*Хорошая таблица сравнения тут: https://www.journaldev.com/21768/junit-vs-testng*